Práctica de Organización del Computador II

Paginación

Segundo Cuatrimestre 2024

Organización del Computador II
DC - UBA

Introducción

Sobre la clase de hoy

En la clase de hoy vamos a ver:

Sobre la clase de hoy

En la clase de hoy vamos a ver:

- **Repaso de paginación como mecanismo**

Sobre la clase de hoy

En la clase de hoy vamos a ver:

- **Repaso de paginación como mecanismo**
- **Proceso de traducción de direcciones virtuales a direcciones físicas**

Sobre la clase de hoy

En la clase de hoy vamos a ver:

- **Repaso de paginación como mecanismo**
- **Proceso de traducción de direcciones virtuales a direcciones físicas**
- **Estructuras involucradas en paginación (`CR3`, `page directory`, `page table`)**

Sobre la clase de hoy

En la clase de hoy vamos a ver:

- **Repaso de paginación como mecanismo**
- **Proceso de traducción de direcciones virtuales a direcciones físicas**
- **Estructuras involucradas en paginación (`CR3`, `page directory`, `page table`)**
- **Atributos relevantes**

Sobre la clase de hoy

En la clase de hoy vamos a ver:

- **Repaso de paginación como mecanismo**
- **Proceso de traducción de direcciones virtuales a direcciones físicas**
- **Estructuras involucradas en paginación (`CR3`, `page directory`, `page table`)**
- **Atributos relevantes**
- **Casos de uso para el mecanismo de paginación**

La idea es presentar la información necesaria para ayudarles a completar el taller.

Repaso de paginación

Paginación en 32 bits

¿Qué significa paginar y qué función cumple la unidad de paginación?

Paginación en 32 bits

¿Qué significa paginar y qué función cumple la unidad de paginación?

- La tarea más relevante desde el punto de vista funcional que cumple la unidad de paginación es la de **traducir una dirección virtual en una dirección física**.

Paginación en 32 bits

¿Qué significa paginar y qué función cumple la unidad de paginación?

- La tarea más relevante desde el punto de vista funcional que cumple la unidad de paginación es la de **traducir una dirección virtual en una dirección física**.
- Esta traducción permite separar el espacio de direcciones expuestas al proceso (dir. virtuales) de su ubicación en la memoria física (dir. físicas en la memoria principal).

Paginación en 32 bits

¿Por qué queremos dividir el espacio de direcciones virtuales del espacio físico?

Paginación en 32 bits

¿Por qué queremos dividir el espacio de direcciones virtuales del espacio físico?

El motivo más relevante tiene que ver con poder separar el espacio de distintos procesos que se ejecutan sobre el mismo procesador y por ende comparten una misma memoria principal.

Paginación en 32 bits

¿Por qué queremos dividir el espacio de direcciones virtuales del espacio físico?

El motivo más relevante tiene que ver con poder separar el espacio de distintos procesos que se ejecutan sobre el mismo procesador y por ende comparten una misma memoria principal.

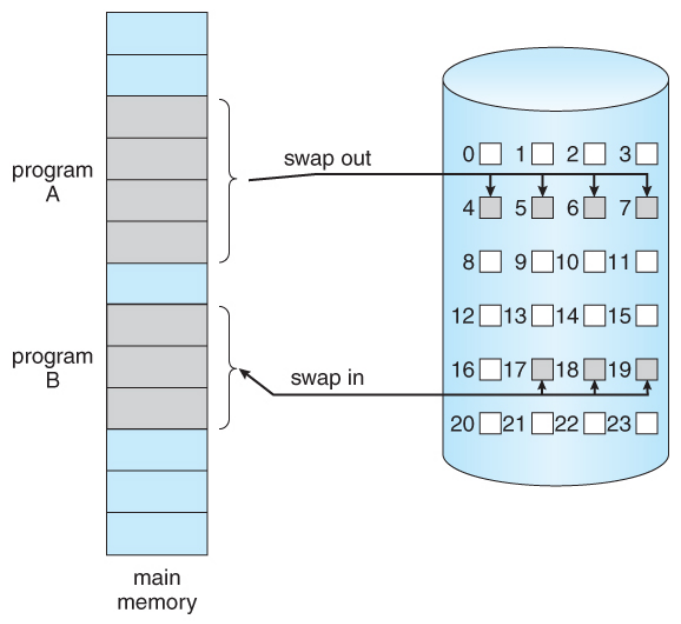
Cada proceso puede manejarse sobre su espacio virtual como si fuera el físico, luego el sistema operativo puede administrar la memoria física como sea necesario, de forma dinámica y sin afectar la ejecución de cada proceso.

Paginación en 32 bits

Lo vamos a ver mejor al cierre de la intro, pero este mecanismo permite por ejemplo que dos procesos utilicen el mismo rango de memoria virtual pero se ubiquen en dos rangos distintos de memoria física, incluso moviéndolos desde el disco a la memoria principal y viceversa.

Paginación en 32 bits

Lo vamos a ver mejor al cierre de la intro, pero este mecanismo permite por ejemplo que dos procesos utilicen el mismo rango de memoria virtual pero se ubiquen en dos rangos distintos de memoria física, incluso moviéndolos desde el disco a la memoria principal y viceversa.



Paginación en 32 bits

La unidad va utilizar dos elementos para realizar la traducción:

Paginación en 32 bits

La unidad va utilizar dos elementos para realizar la traducción:

- **La dirección virtual** que es la que conoce el proceso.

Paginación en 32 bits

La unidad va utilizar dos elementos para realizar la traducción:

- **La dirección virtual** que es la que conoce el proceso.
- **La estructura de paginación (directorio y tablas de páginas) del proceso** qué indican qué direcciones virtuales se corresponden a qué direcciones físicas.

Paginación en 32 bits

La ubicación del directorio de páginas se encuentra en el registro de control `CR3`, con lo cual si la dirección virtual a resolver es `virt` y su dirección física asociada es `phys`, podemos definir el proceso de traducción implementado por la unidad de paginación como:

Paginación en 32 bits

La ubicación del directorio de páginas se encuentra en el registro de control `CR3`, con lo cual si la dirección virtual a resolver es `virt` y su dirección física asociada es `phys`, podemos definir el proceso de traducción implementado por la unidad de paginación como:

- $map : uint32 \times CR3 \rightarrow uint32$

Paginación en 32 bits

La ubicación del directorio de páginas se encuentra en el registro de control `CR3`, con lo cual si la dirección virtual a resolver es `virt` y su dirección física asociada es `phys`, podemos definir el proceso de traducción implementado por la unidad de paginación como:

- $map : uint32 \times CR3 \rightarrow uint32$
- $map(virt, CR3) = phys$.

Traduciendo direcciones

Traducción de direcciones virtuales

Veamos ahora cómo se traducen las direcciones virtuales a partir de un valor de dirección virtual y un directorio de páginas.

Traducción de direcciones virtuales

Veamos ahora cómo se traducen las direcciones virtuales a partir de un valor de dirección virtual y un directorio de páginas.
Es importante notar que vamos a observar la dirección virtual y **dividirla en tres partes**.

Traducción de direcciones virtuales

Veamos ahora cómo se traducen las direcciones virtuales a partir de un valor de dirección virtual y un directorio de páginas.
Es importante notar que vamos a observar la dirección virtual y **dividirla en tres partes**.

- **Los 10 bits más altos** indicarán en qué entrada del directorio vamos buscar la ubicación de la tabla de páginas necesaria (pd_index).

Traducción de direcciones virtuales

Veamos ahora cómo se traducen las direcciones virtuales a partir de un valor de dirección virtual y un directorio de páginas.
Es importante notar que vamos a observar la dirección virtual y **dividirla en tres partes**.

- **Los 10 bits más altos** indicarán en qué entrada del directorio vamos buscar la ubicación de la tabla de páginas necesaria (pd_index).
- **Los 10 bits siguientes** indicarán en qué entrada de la tabla vamos buscar la ubicación de la dirección base de la página necesaria (pt_index).

Traducción de direcciones virtuales

Veamos ahora cómo se traducen las direcciones virtuales a partir de un valor de dirección virtual y un directorio de páginas.
Es importante notar que vamos a observar la dirección virtual y **dividirla en tres partes**.

- **Los 10 bits más altos** indicarán en qué entrada del directorio vamos buscar la ubicación de la tabla de páginas necesaria (*pd_index*).
- **Los 10 bits siguientes** indicarán en qué entrada de la tabla vamos buscar la ubicación de la dirección base de la página necesaria (*pt_index*).
- **Los 12 bits más bajos** indicarán el desplazamiento desde la base de la página donde se encuentra el dato esperado (*page_offset*).

Mecanismo de Paginación

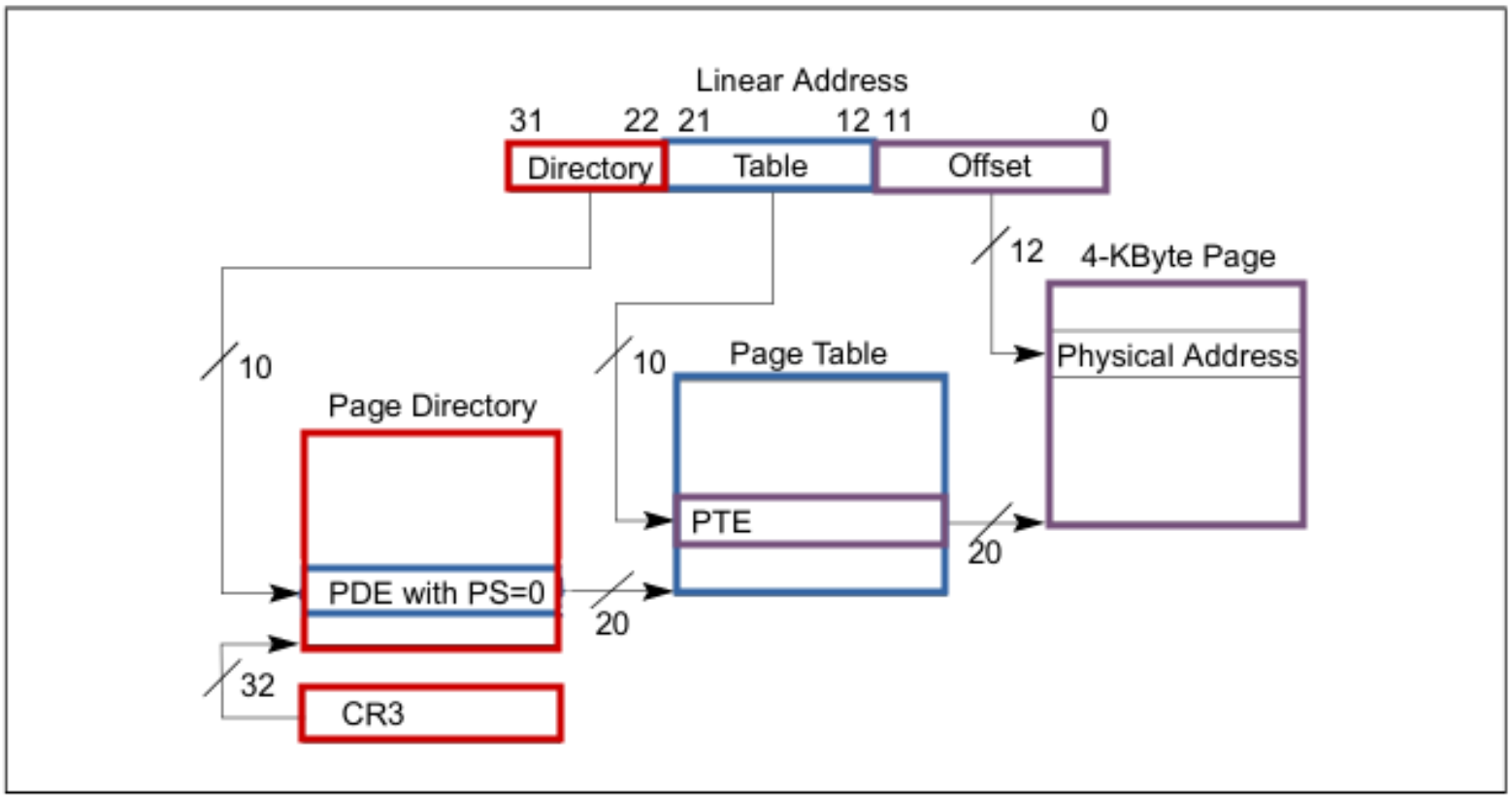


Figure 4-2. Linear-Address Translation to a 4-KByte Page using 32-Bit Paging

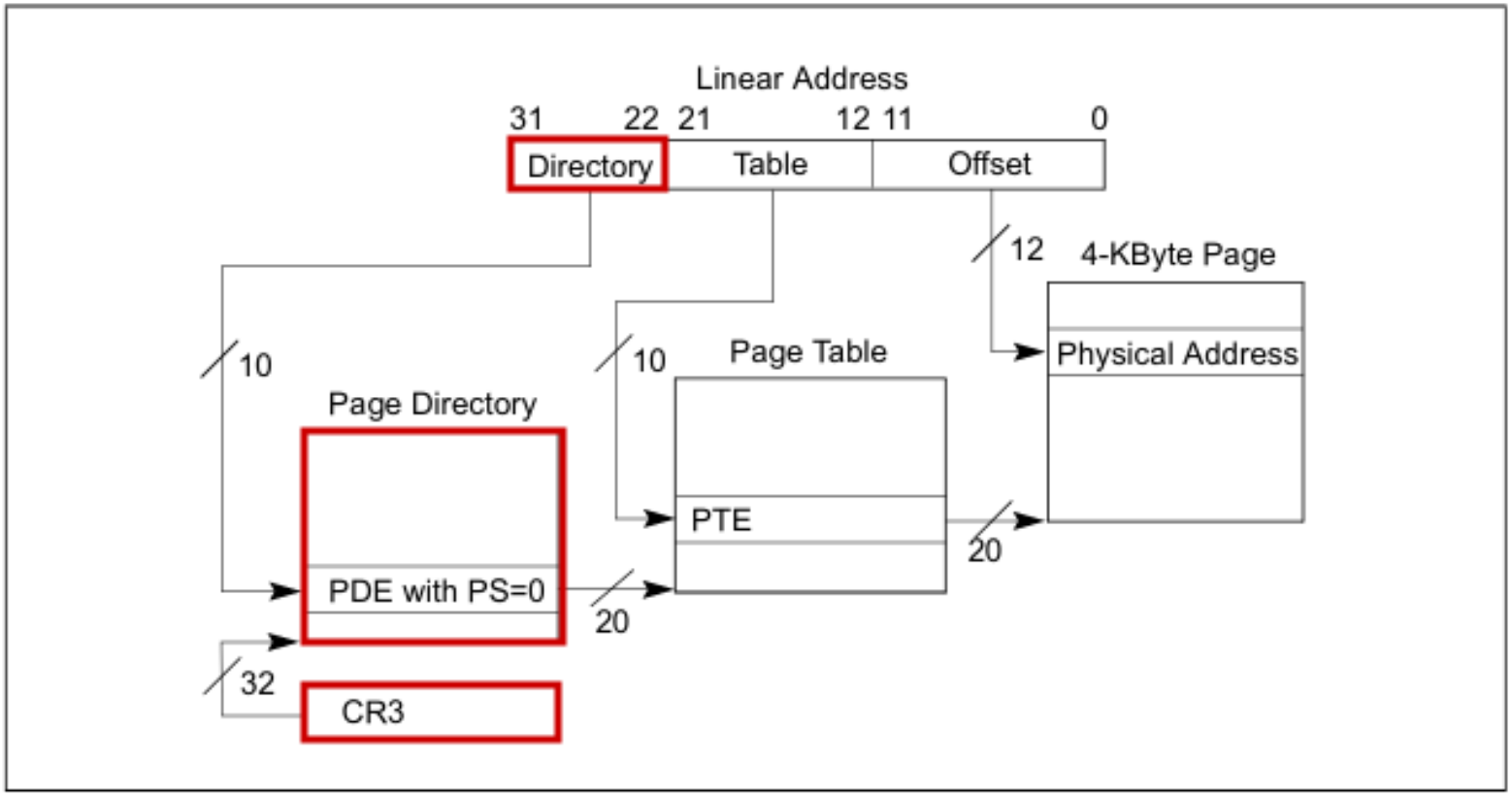


Figure 4-2. Linear-Address Translation to a 4-KByte Page using 32-Bit Paging

Veamos primero el CR3.

CR3

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Address of page directory[1]																				Ignored							PCD	PWT	Ignored				**CR3**

CR3

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Address of page directory[1]																				Ignored							PCD	PWT	Ignored				**CR3**

La parte más importante del registro de control `CR3` son los 20 bits más altos, ya que contienen la dirección de la página donde se encuentra el directorio a utilizar para traducir las direcciones virtuales.

Dirección del directorio: $CR3\&0xFFFFF000$

Directorio de páginas

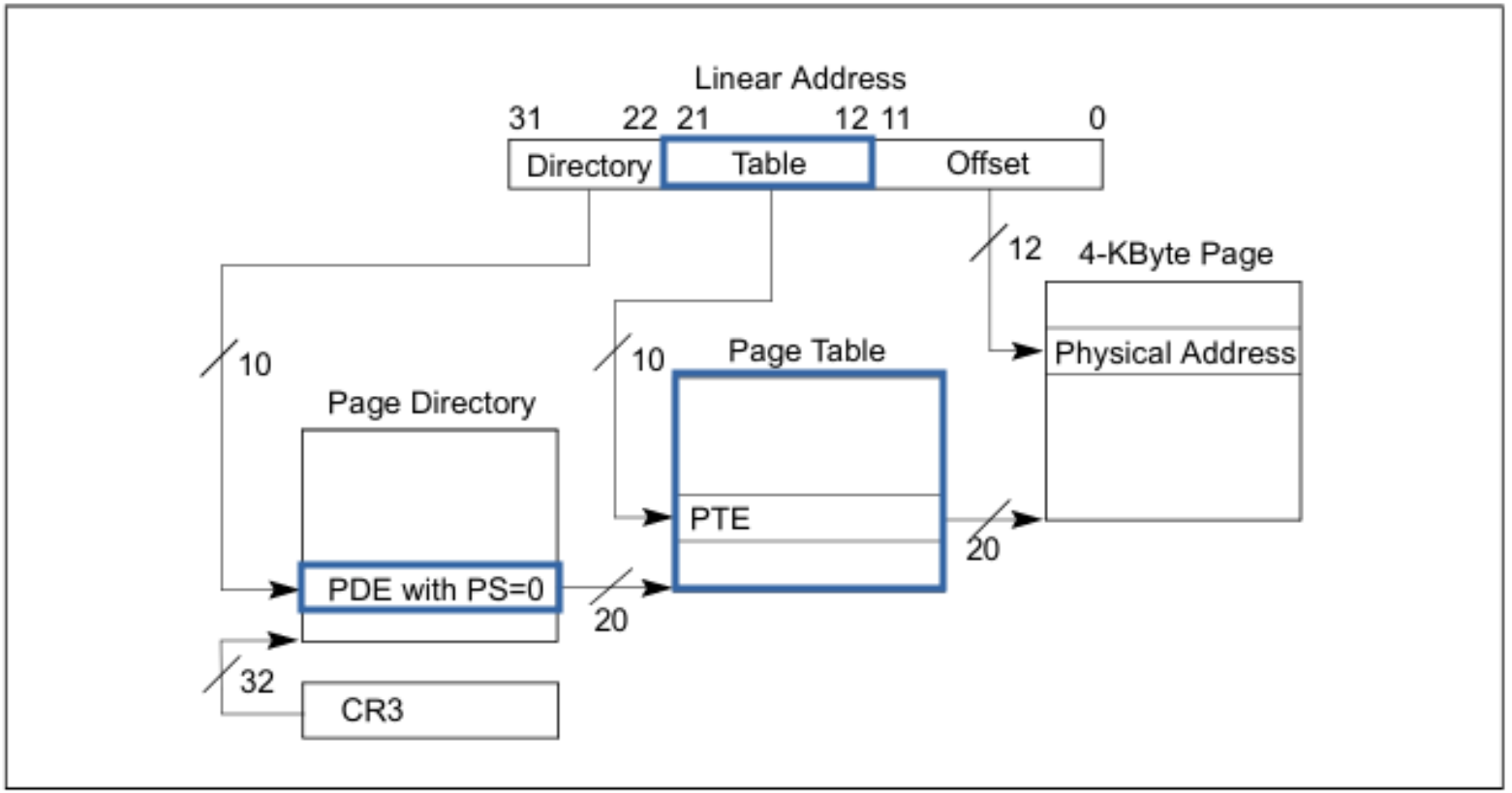


Figure 4-2. Linear-Address Translation to a 4-KByte Page using 32-Bit Paging

Ahora veamos las entradas (**PDE** por page directory entry) del directorio de páginas (**PD** por page directory).

Directorio de páginas

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Address of page table																				Ignored				**0**	Ign	A	PCD	PWT	U/S	R/W	**1**	**PDE: page table**

Directorio de páginas

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Address of page table																				Ignored				**0**	Ign	A	PCD	PWT	U/S	R/W	**1**	**PDE: page table**

Tanto el directorio como la tabla de páginas tendrán 1024 entradas, como cada entrada es de 32 bits (4 bytes) el directorio y las tablas ocupan 1 página (4KByte). Arriba vemos la estructura de una entrada del directorio. Los 20 bits más altos de la i-ésima entrada corresponden a la dirección de la i-ésima tabla de páginas.

Dirección de la i-ésima tabla: $\boxed{pd[i]\&0xFFFFF000}$

Tabla de páginas

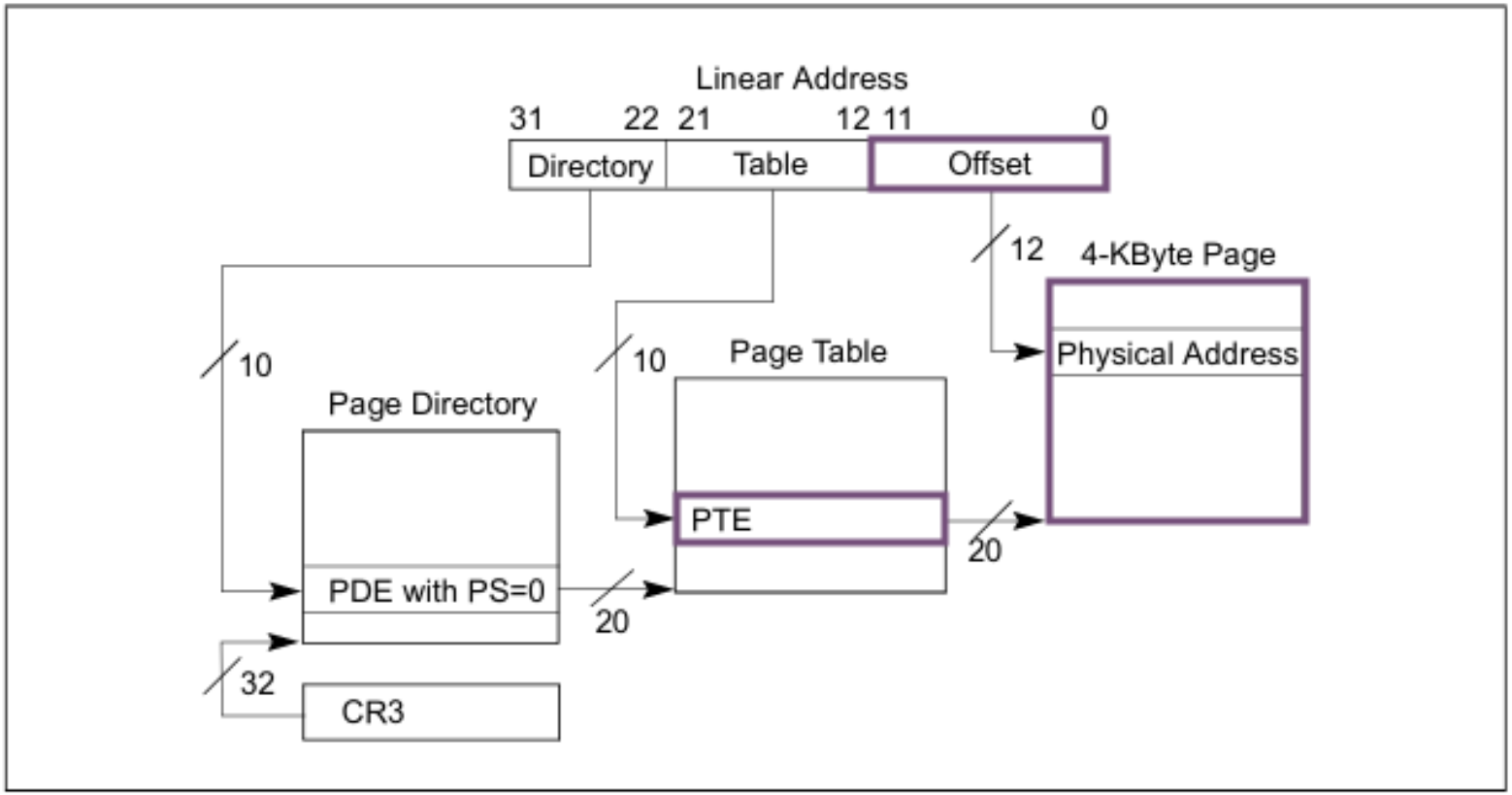


Figure 4-2. Linear-Address Translation to a 4-KByte Page using 32-Bit Paging

Ahora veamos la tabla de páginas (**PT** por page table) y sus entradas (**PTE** por page table entry).

Tabla de páginas

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Address of 4KB page frame																				Ignored			G	PAT	D	A	PCD	PWT	U/S	R/W	1	**PTE: 4KB page**

Tabla de páginas

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Address of 4KB page frame																				Ignored			G	PAT	D	A	PCD	PWT	U/S	R/W	1	**PTE: 4KB page**

Arriba vemos la estructura de una entrada de la tabla. Los 20 bits más altos de la i-ésima entrada corresponden a la dirección de la i-ésima página.

Dirección de la i-ésima página: $pt[i]\&0xFFFFF000$

Operaciones de la traducción

Queremos traducir la dirección

$$\boxed{virt = dir(10bits)|table(10bits)|offset(12bits)}.$$

Operaciones de la traducción

Queremos traducir la dirección

$$\boxed{virt = dir(10bits)|table(10bits)|offset(12bits)}.$$

- $\boxed{pd := CR3\&0xFFFFF000}$ Dirección de PD (limpiamos CR3).

Operaciones de la traducción

Queremos traducir la dirección

$\boxed{virt = dir(10bits)|table(10bits)|offset(12bits)}$.

- $\boxed{pd := CR3\&0xFFFFF000}$ Dirección de PD (limpiamos CR3).
- $\boxed{pd\_index := (virt >> 22)\&0x3FF}$ Índice de PD (los 10 bits más altos de $virt$).

Operaciones de la traducción

Queremos traducir la dirección
$\boxed{virt = dir(10bits)|table(10bits)|offset(12bits)}$.

- $\boxed{pd := CR3\&0xFFFFF000}$ Dirección de PD (limpiamos CR3).
- $\boxed{pd\_index := (virt >> 22)\&0x3FF}$ Índice de PD (los 10 bits más altos de $virt$).
- $\boxed{pt := pd[pd\_index]\&0xFFFFF000}$ Dirección de PT (limpiamos la PDE).

Operaciones de la traducción

Queremos traducir la dirección

$$virt = dir(10bits)|table(10bits)|offset(12bits).$$

- $pd := CR3\&0xFFFFF000$ Dirección de PD (limpiamos CR3).
- $pd\_index := (virt >> 22)\&0x3FF$ Índice de PD (los 10 bits más altos de *virt*).
- $pt := pd[pd\_index]\&0xFFFFF000$ Dirección de PT (limpiamos la PDE).
- $pt\_index := (virt >> 12)\&0x3FF$ Índice de PT (los 10 bits del medio de *virt*).

Operaciones de la traducción

Queremos traducir la dirección

$$virt = dir(10bits)|table(10bits)|offset(12bits).$$

- $pd := CR3\&0xFFFFF000$ Dirección de PD (limpiamos CR3).
- $pd\_index := (virt >> 22)\&0x3FF$ Índice de PD (los 10 bits más altos de *virt*).
- $pt := pd[pd\_index]\&0xFFFFF000$ Dirección de PT (limpiamos la PDE).
- $pt\_index := (virt >> 12)\&0x3FF$ Índice de PT (los 10 bits del medio de *virt*).
- $page\_addr := pt[pt\_index]\&0xFFFFF000$ Dirección de la página (limpiamos la PTE).

Operaciones de la traducción

Queremos traducir la dirección

$$virt = dir(10bits)|table(10bits)|offset(12bits).$$

- $pd := CR3\&0xFFFFF000$ Dirección de PD (limpiamos CR3).
- $pd\_index := (virt >> 22)\&0x3FF$ Índice de PD (los 10 bits más altos de *virt*).
- $pt := pd[pd\_index]\&0xFFFFF000$ Dirección de PT (limpiamos la PDE).
- $pt\_index := (virt >> 12)\&0x3FF$ Índice de PT (los 10 bits del medio de *virt*).
- $page\_addr := pt[pt\_index]\&0xFFFFF000$ Dirección de la página (limpiamos la PTE).
- $offset := virt\&0xFFF$ Offset desde el inicio de la página (los 12 bits más bajos de *virt*).

Operaciones de la traducción

Queremos traducir la dirección

$$virt = dir(10bits)|table(10bits)|offset(12bits).$$

- $pd := CR3\&0xFFFFF000$ Dirección de PD (limpiamos CR3).
- $pd\_index := (virt >> 22)\&0x3FF$ Índice de PD (los 10 bits más altos de *virt*).
- $pt := pd[pd\_index]\&0xFFFFF000$ Dirección de PT (limpiamos la PDE).
- $pt\_index := (virt >> 12)\&0x3FF$ Índice de PT (los 10 bits del medio de *virt*).
- $page\_addr := pt[pt\_index]\&0xFFFFF000$ Dirección de la página (limpiamos la PTE).
- $offset := virt\&0xFFF$ Offset desde el inicio de la página (los 12 bits más bajos de *virt*).
- $phys := page\_addr|offset$ Dirección física (sumamos la base de la página y el offset de *virt*).

Taller

Taller

En el taller van a tener que implementar algunas funciones relacionadas con la paginación y la unidad de manejo de memoria.

```
paddr_t mmu_next_free_kernel_page(void);
paddr_t mmu_next_free_user_page(void);
```

- Devuelve la dirección física de la próxima página de kernel/user disponible.
- Las páginas de kernel y de usuario se encuentran en rangos de direcciones distintos, por lo que es necesario llevar un contador para cada uno.

Taller

```
paddr_t mmu_init_kernel_dir(void);
```

- Genera el identity mapping de las primeras 4MB de memoria.
- Recuerden setear correctamente los atributos de las páginas (W, S, P).
- ¿Cuántas páginas necesito para armar las tablas?

Taller

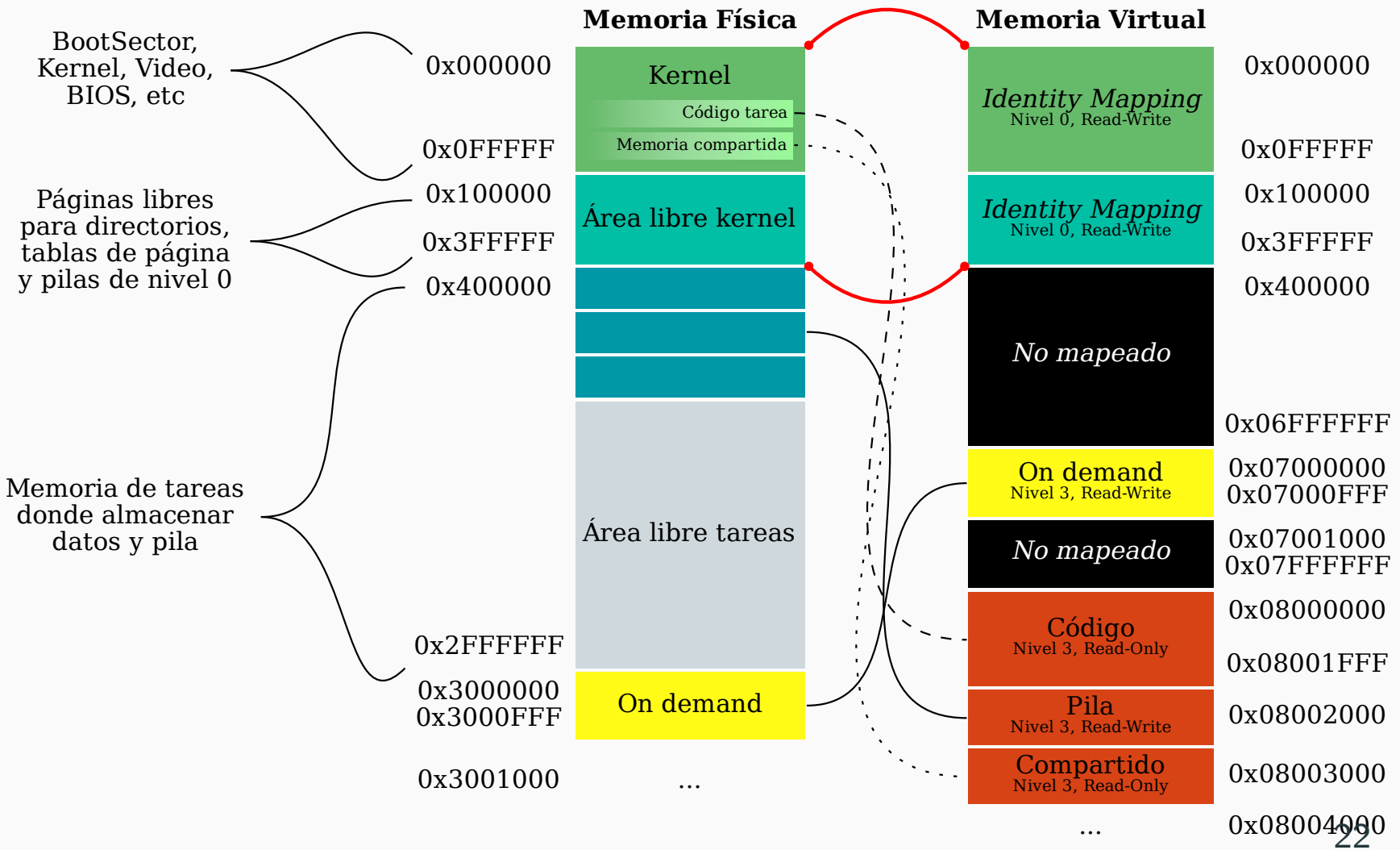

```
void mmu_map_page(uint32_t cr3, vaddr_t virt, paddr_t phy,
↪  uint32_t attrs);
```

- Genera un mapeo de una página virtual a una página física.
- ¿en qué directorio? ¿Y si ya existe la page table?
- ¿Los atributos van para la PDE o la PTE?
- Recuerden llamar a `tlbflush`

Taller


```
paddr_t mmu_unmap_page(uint32_t cr3, vaddr_t virt);
```

- Desmapea una página virtual y devuelve la dirección de la página física desmapeada
- Pueden preguntar por Present y actuar en consecuencia
- Recuerden llamar a `tlbflush`


Memoria Física
Memoria Virtual
BootSector, Kernel, Video, BIOS, etc
Páginas libres para directorios, tablas de página y pilas de nivel 0
Memoria de tareas donde almacenar datos y pila
0x000000
0x0FFFFF
0x100000
0x3FFFFF
0x400000
0x2FFFFFF
0x3000000
0x3000FFF
0x3001000
Kernel
Código tarea
Memoria compartida
Área libre kernel
Área libre tareas
On demand
...
Identity Mapping
Nivel 0, Read-Write
Identity Mapping
Nivel 0, Read-Write
No mapeado
0x06FFFFFF
On demand
Nivel 3, Read-Write
0x07000000
0x07000FFF
No mapeado
0x07001000
0x07FFFFFF
Código
Nivel 3, Read-Only
0x08000000
0x08001FFF
Pila
Nivel 3, Read-Write
0x08002000
Compartido
Nivel 3, Read-Only
0x08003000
...
0x08004000

Taller

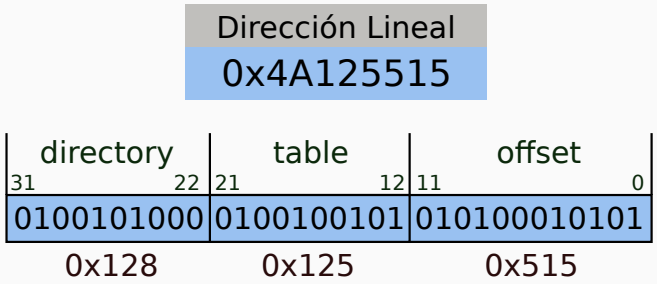

```
paddr_t mmu_init_task_dir(paddr_t phy_start)
```

- Genera el mapeo estático de una tarea del sistema
- Mapea 2 páginas de código, una de pila y una shared
- Recuerden mapear al kernel en los primeros 4MB

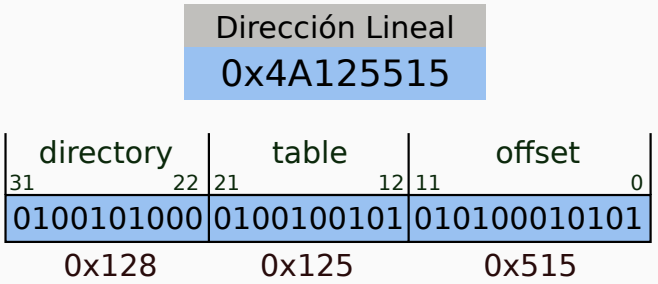
Ejemplo Numérico

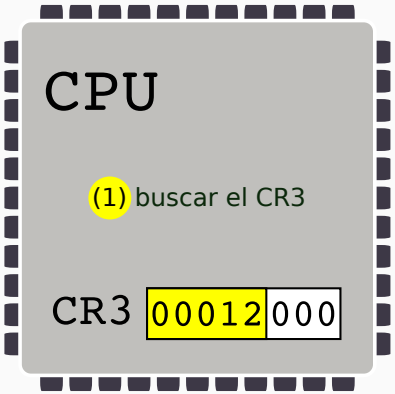
Dirección Lineal
0x4A125515

Ejemplo Numérico

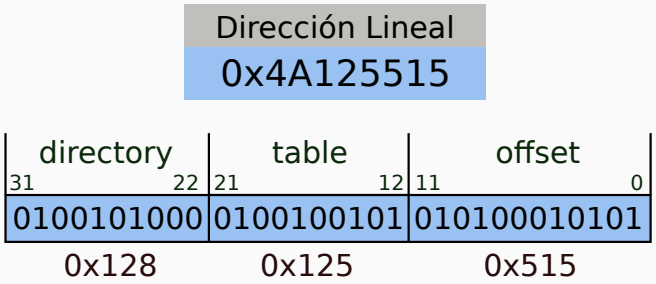


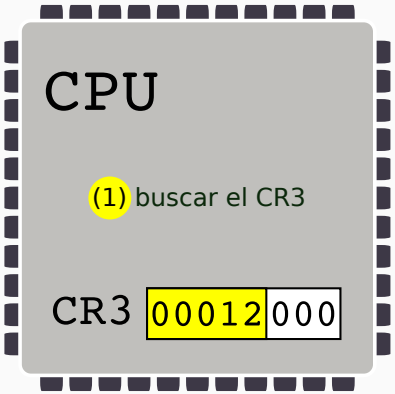
Ejemplo Numérico

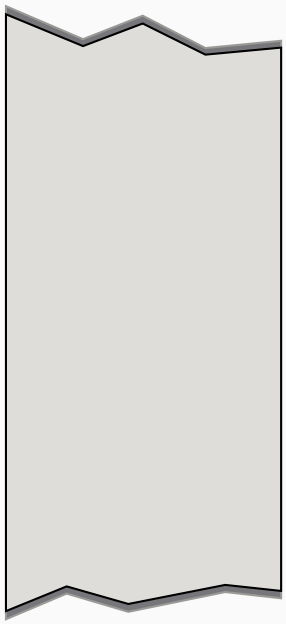




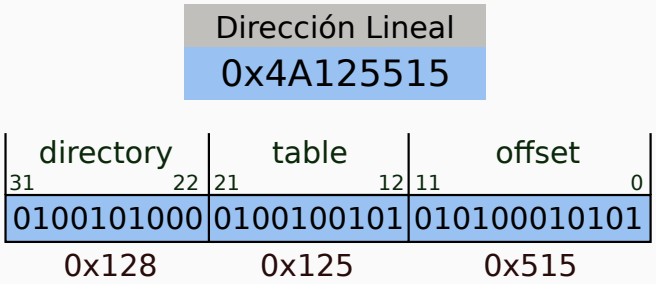
Ejemplo Numérico

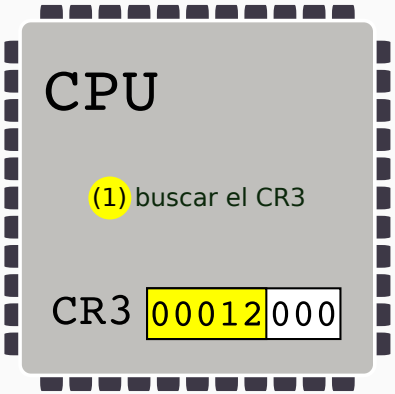


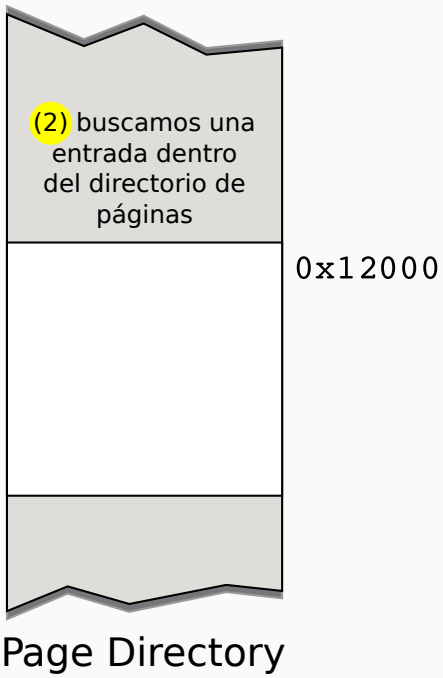




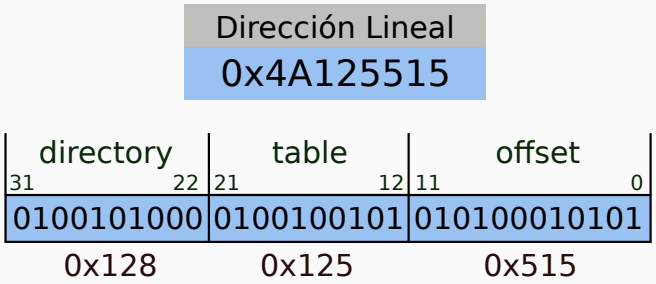
Ejemplo Numérico

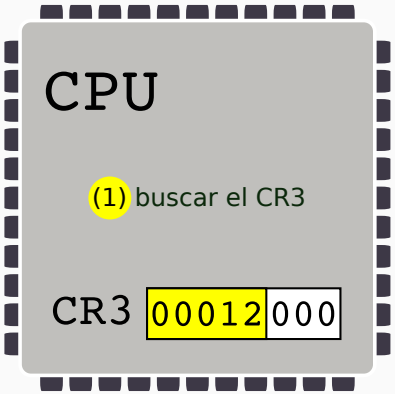


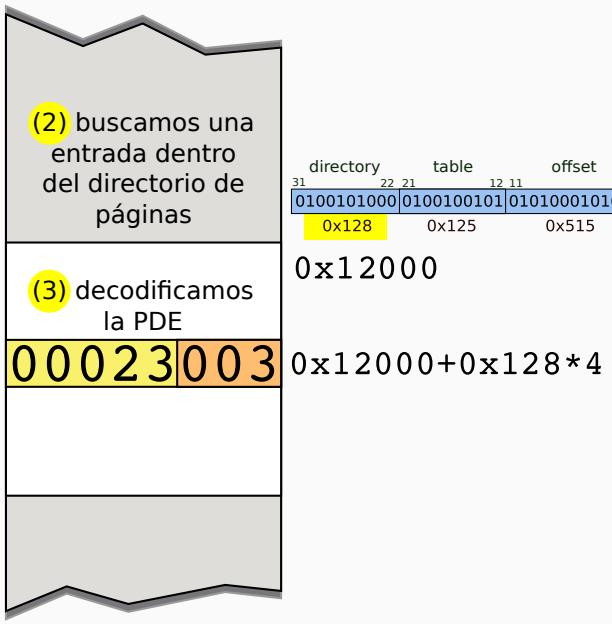




Ejemplo Numérico







0x12000

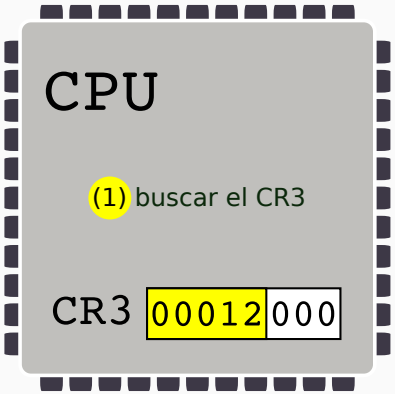
0x12000+0x128*4

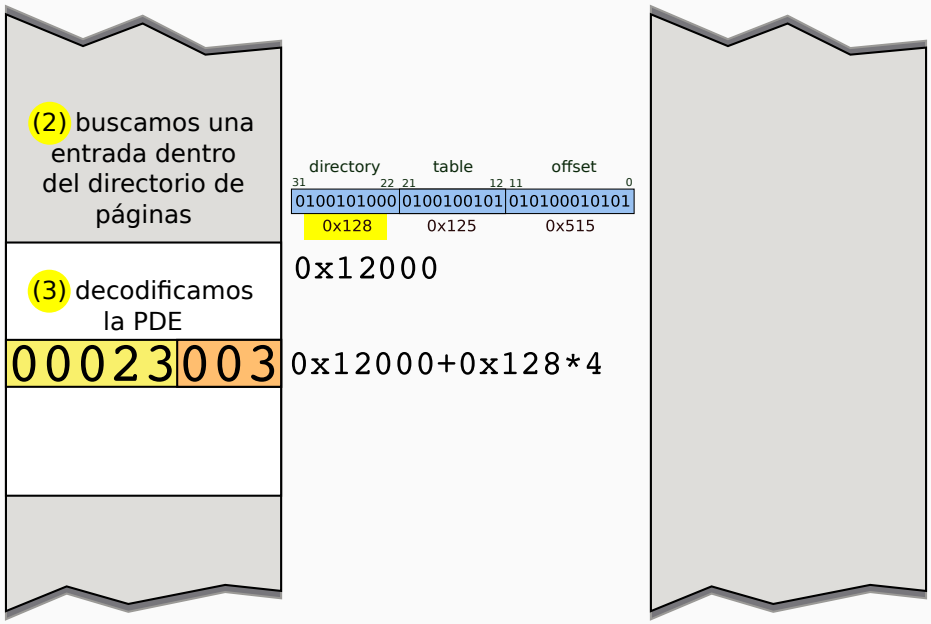
Ejemplo Numérico

Dirección Lineal

0x4A125515

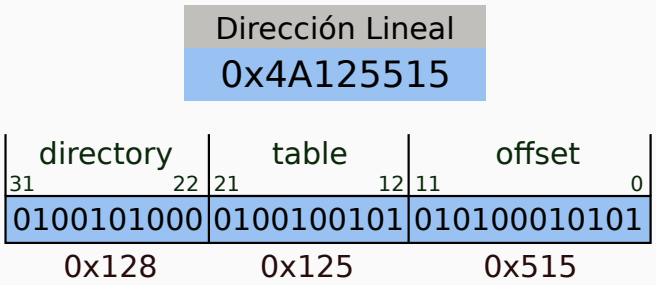
directory	table	offset
31 22	21 12	11 0
0100101000	0100100101	010100010101
0x128	0x125	0x515

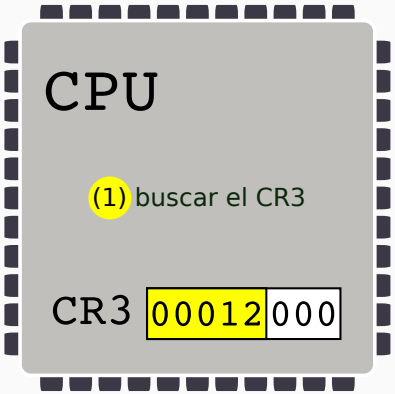


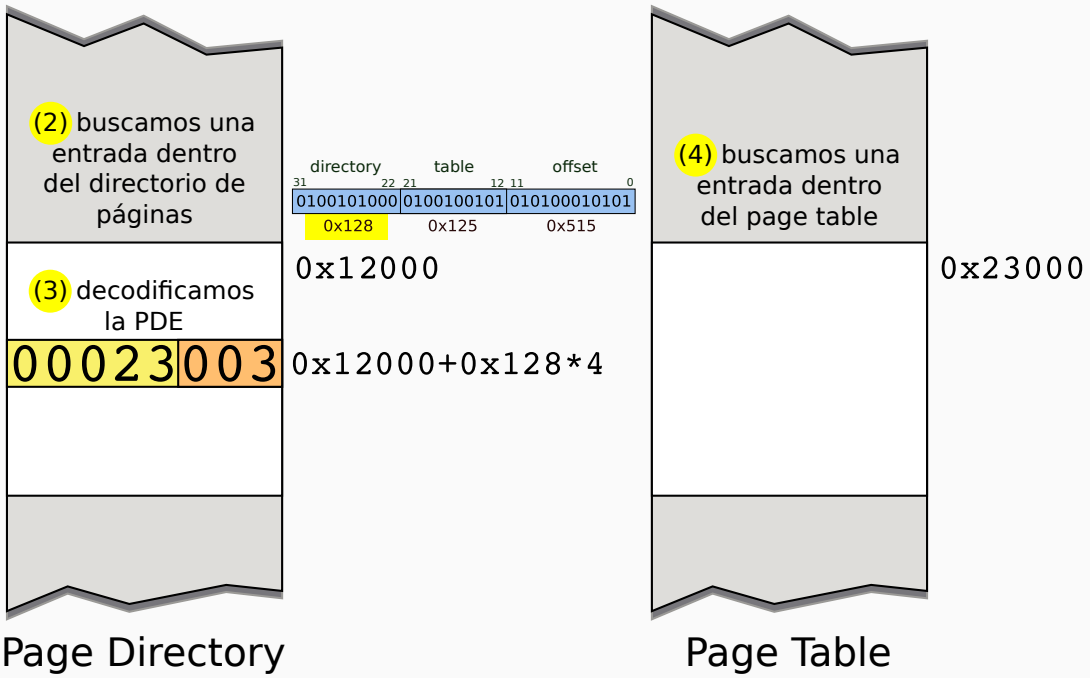


Page Directory

Ejemplo Numérico

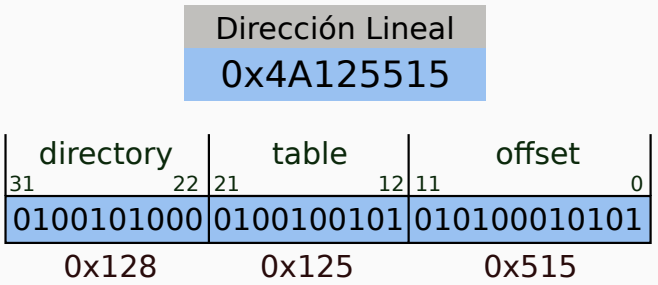


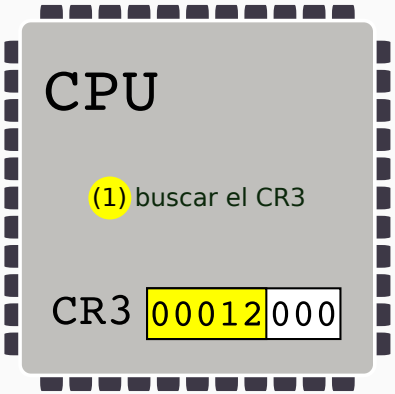


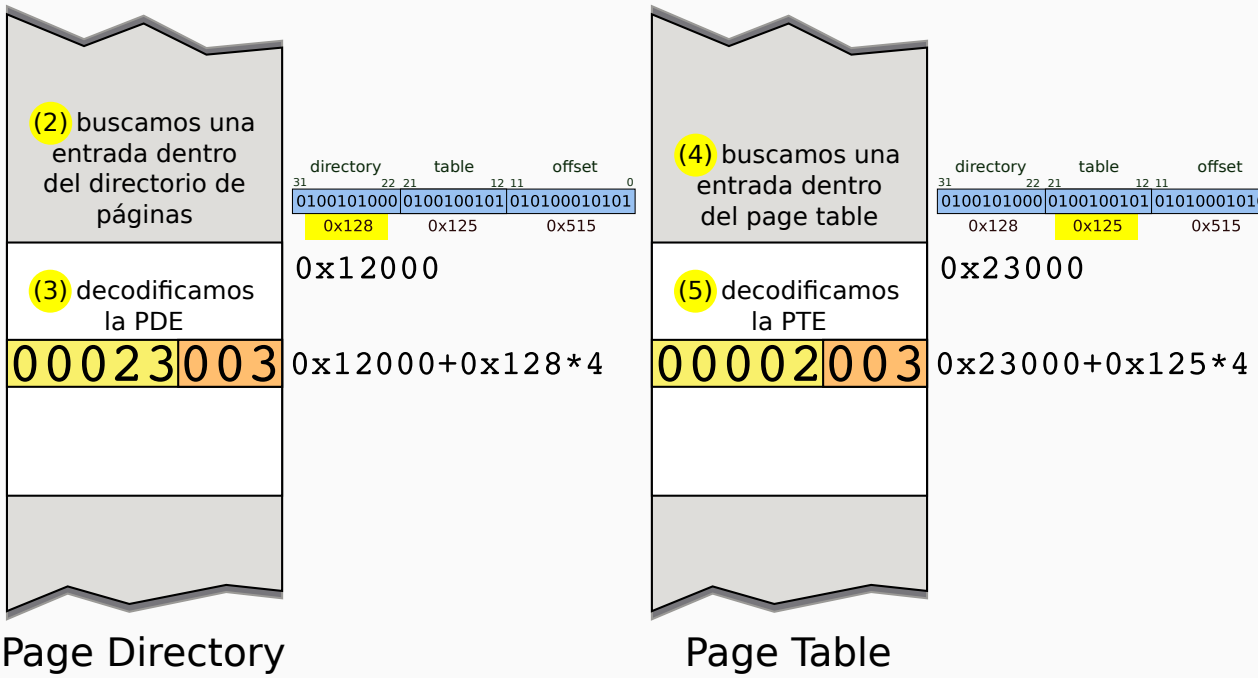


Ejemplo Numérico

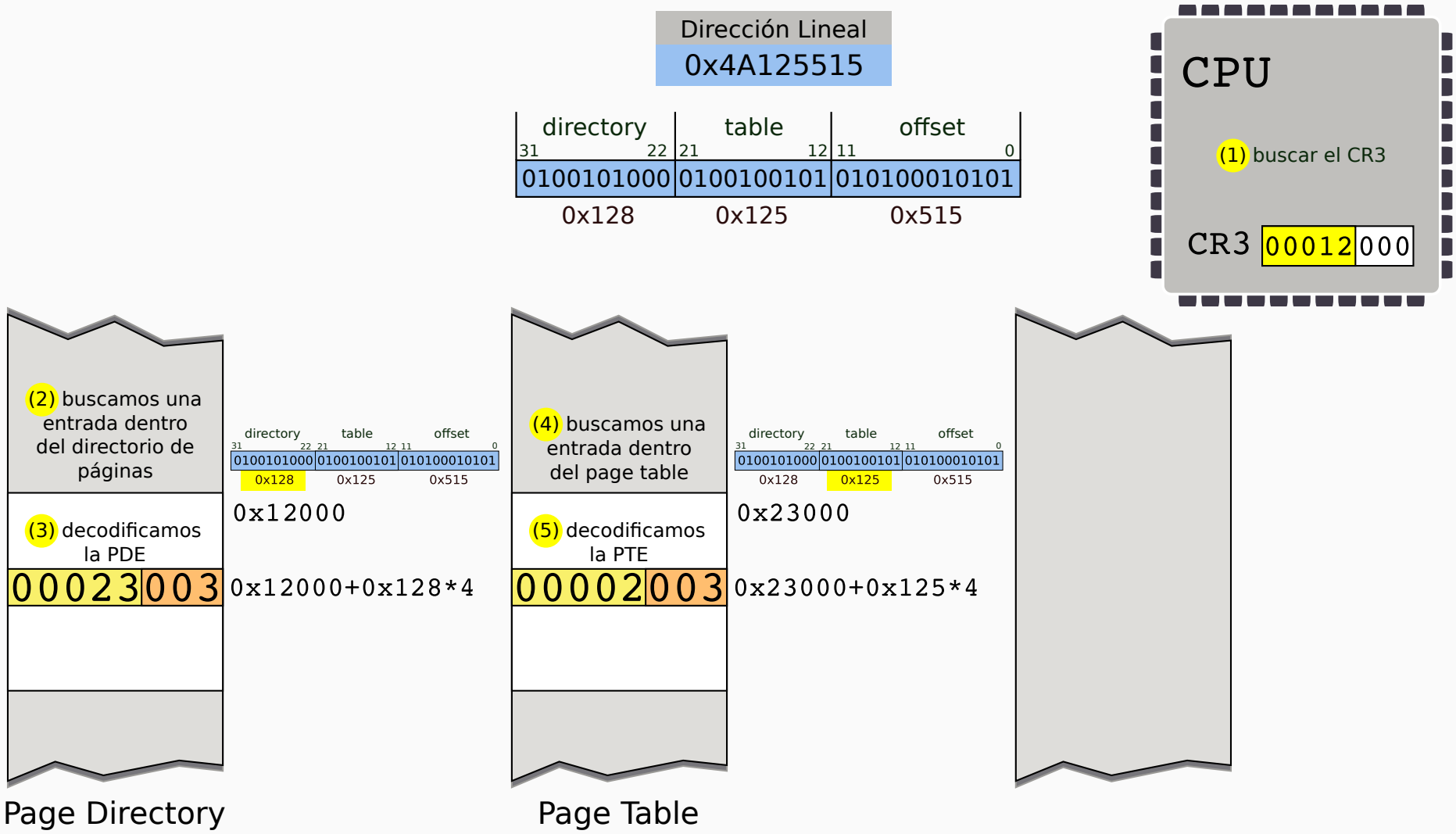
DEPARTAMENTO DE COMPUTACION
Facultad de Ciencias Exactas y Naturales - UBA



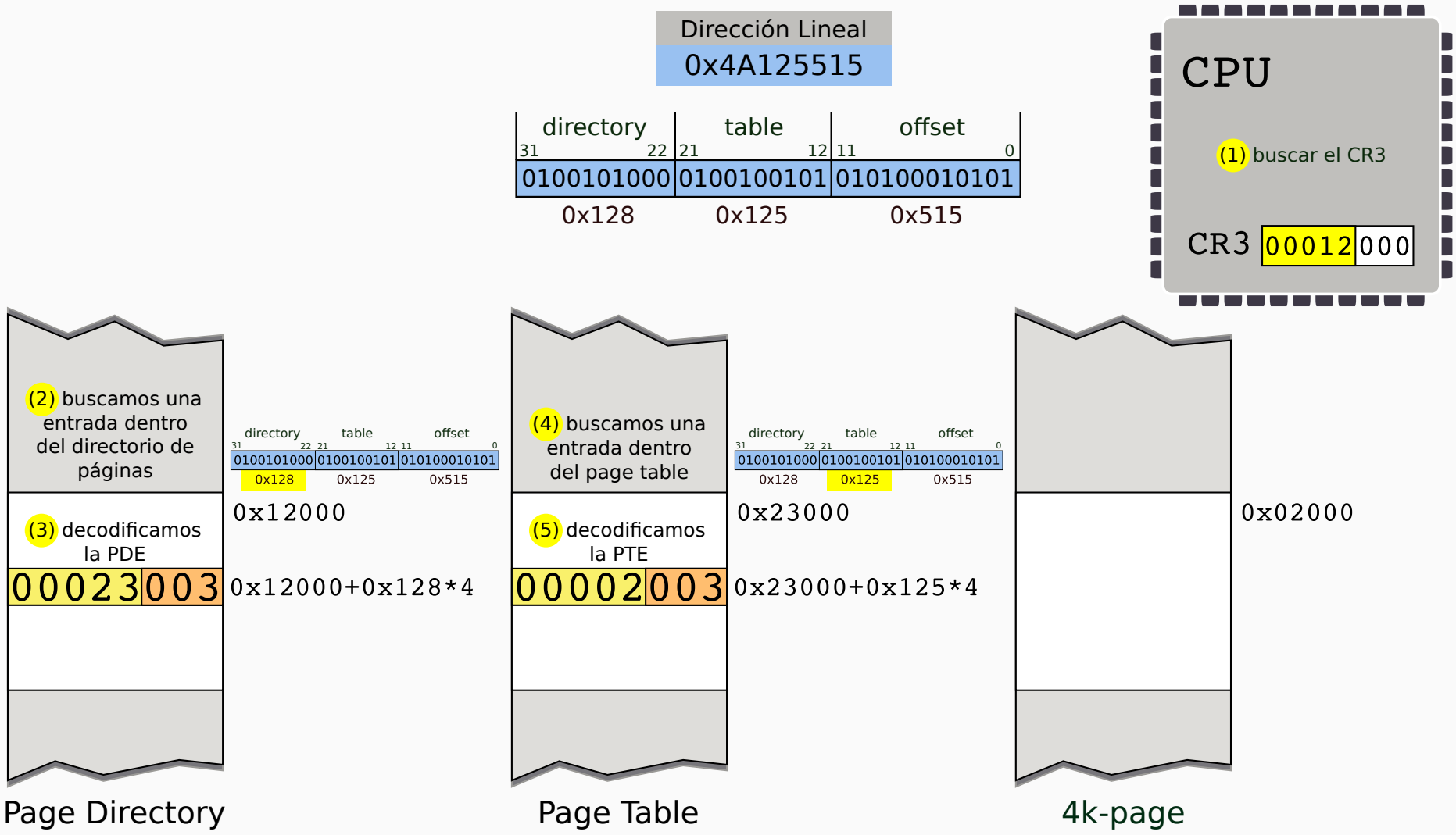




Ejemplo Numérico

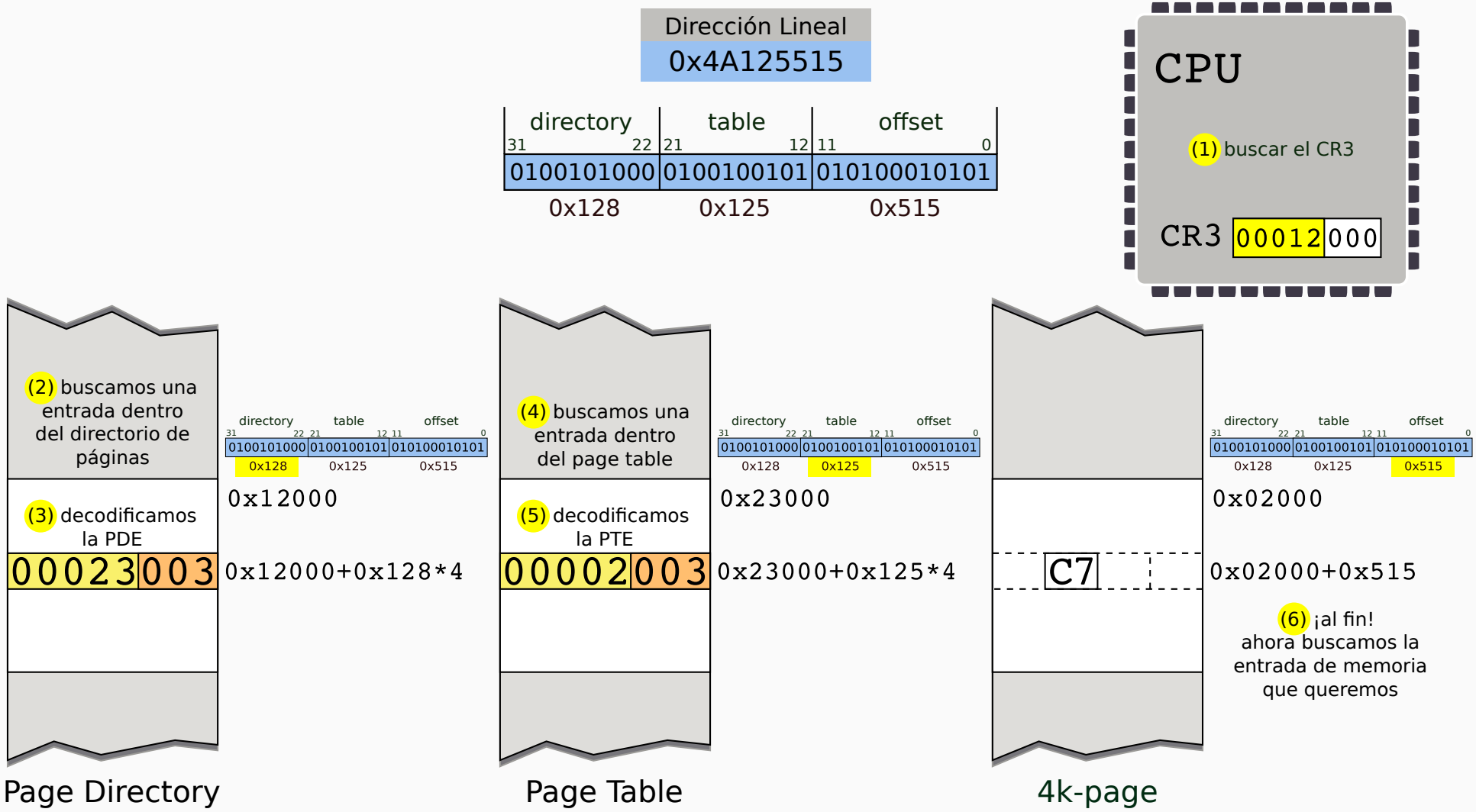


Ejemplo Numérico



Ejemplo Numérico

DEPARTAMENTO DE COMPUTACION
Facultad de Ciencias Exactas y Naturales - UBA



TLB (translation lookaside buffer)

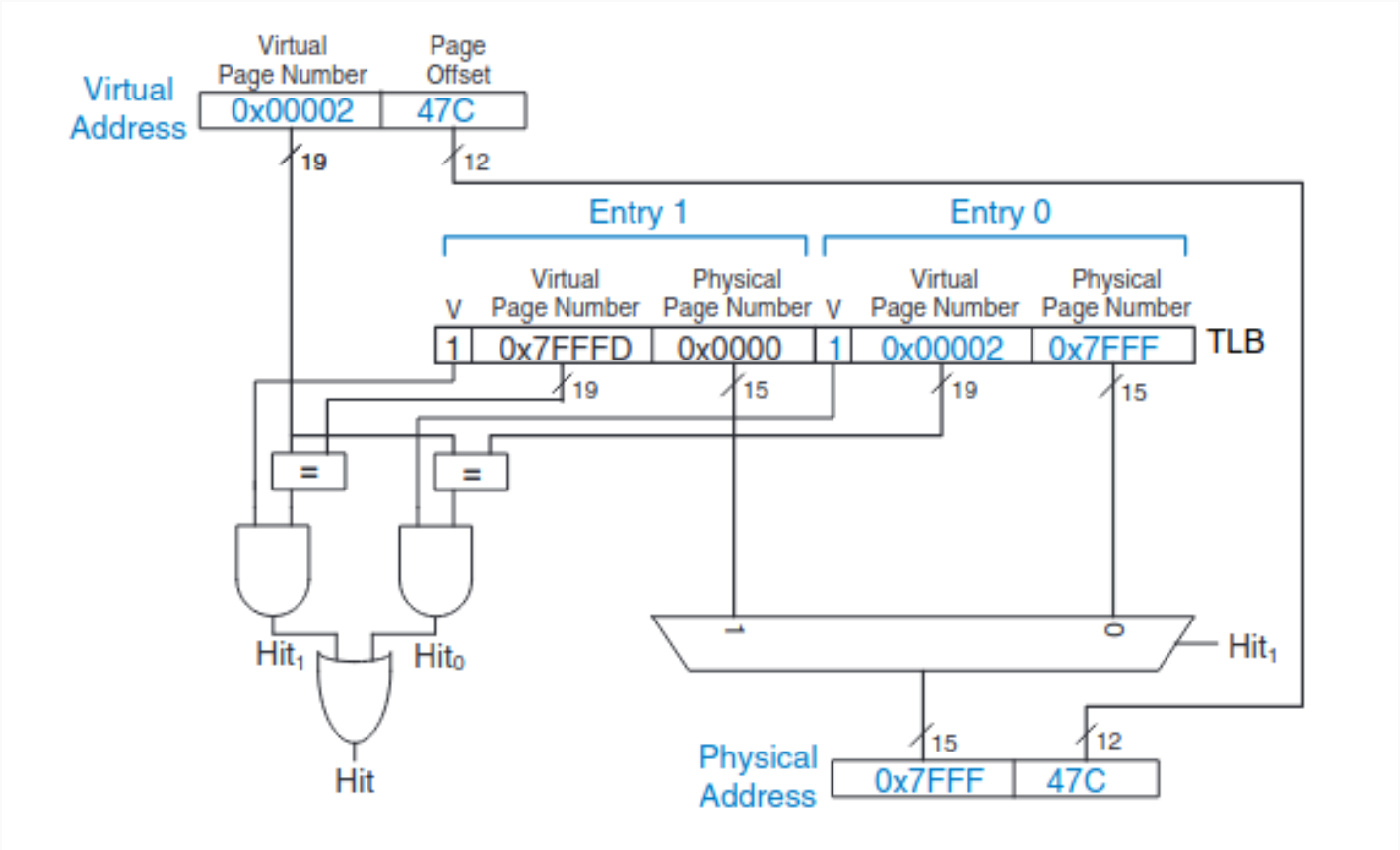
Recuerden que el procesador cuenta con una tabla de traducciones pre-computadas a la que llamaremos buffer auxiliar de traducción o **translation lookaside buffer**, que básicamente almacena las últimas traducciones realizadas para no tener que volver a computarlas.

TLB (translation lookaside buffer)

Recuerden que el procesador cuenta con una tabla de traducciones pre-computadas a la que llamaremos buffer auxiliar de traducción o **translation lookaside buffer**, que básicamente almacena las últimas traducciones realizadas para no tener que volver a computarlas.

Cuando realicemos un cambio en nuestras estructuras de paginación es necesario forzar una limpieza del mismo para evitar que las direcciones pre-computadas que ya no son válidas se sigan empleando, para esto realizamos un intercambio del registro `CR3` con un valor temporal y luego lo restauramos.

Ejemplo de uso de la TLB



Atributos de la PTE

Veamos qué atributos se encuentran disponibles en las entradas de la tabla de páginas para comprender cómo se implementan algunos mecanismos que presentaremos en breve.

Atributos de la PTE

Veamos qué atributos se encuentran disponibles en las entradas de la tabla de páginas para comprender cómo se implementan algunos mecanismos que presentaremos en breve.
Recuerden que a diferencia de lo que sucede con segmentación en paginación sólo tenemos **dos niveles de privilegio, supervisor o kernel (0) y usuarix (1)**.

Atributos de la PTE

- El bit **D**(dirty) es seteado por unidad de memoria del procesador cuando se escribe en la página, se limpia por soft.

Atributos de la PTE

- El bit **D**(dirty) es seteado por unidad de memoria del procesador cuando se escribe en la página, se limpia por soft.
- El bit **A**(accessed) es seteado por unidad de memoria del procesador cuando se escribe o lee en la página, se limpia por soft.

Atributos de la PTE

- El bit **D**(dirty) es seteado por unidad de memoria del procesador cuando se escribe en la página, se limpia por soft.
- El bit **A**(accessed) es seteado por unidad de memoria del procesador cuando se escribe o lee en la página, se limpia por soft.
- El **PCD** (cache disabled) hace que la página no se almacene en memoria rápida.

Atributos de la PTE

- El bit **D**(dirty) es seteado por unidad de memoria del procesador cuando se escribe en la página, se limpia por soft.
- El bit **A**(accessed) es seteado por unidad de memoria del procesador cuando se escribe o lee en la página, se limpia por soft.
- El **PCD** (cache disabled) hace que la página no se almacene en memoria rápida.
- **PWT** (write through) hace que al escribir la escritura se refleje en cache y memoria a la vez, sino sólo se actualiza la memoria cuando se desaloja la línea de caché.

Atributos de la PTE

- **U/S** indica si la página puede ser accedida por el espacio de usuarix (bit en 1) o sólo supervisor/kernel (bit en 0).

Atributos de la PTE

- **U/S** indica si la página puede ser accedida por el espacio de usuarix (bit en 1) o sólo supervisor/kernel (bit en 0).
- **R/W** indica si la página puede leerse y escribirse (bit en 1) o sólo leerse (bit en 0).

Atributos de la PTE

- **U/S** indica si la página puede ser accedida por el espacio de usuarix (bit en 1) o sólo supervisor/kernel (bit en 0).
- **R/W** indica si la página puede leerse y escribirse (bit en 1) o sólo leerse (bit en 0).
- **P** indica si la página se encuentra cargada en memoria o no.

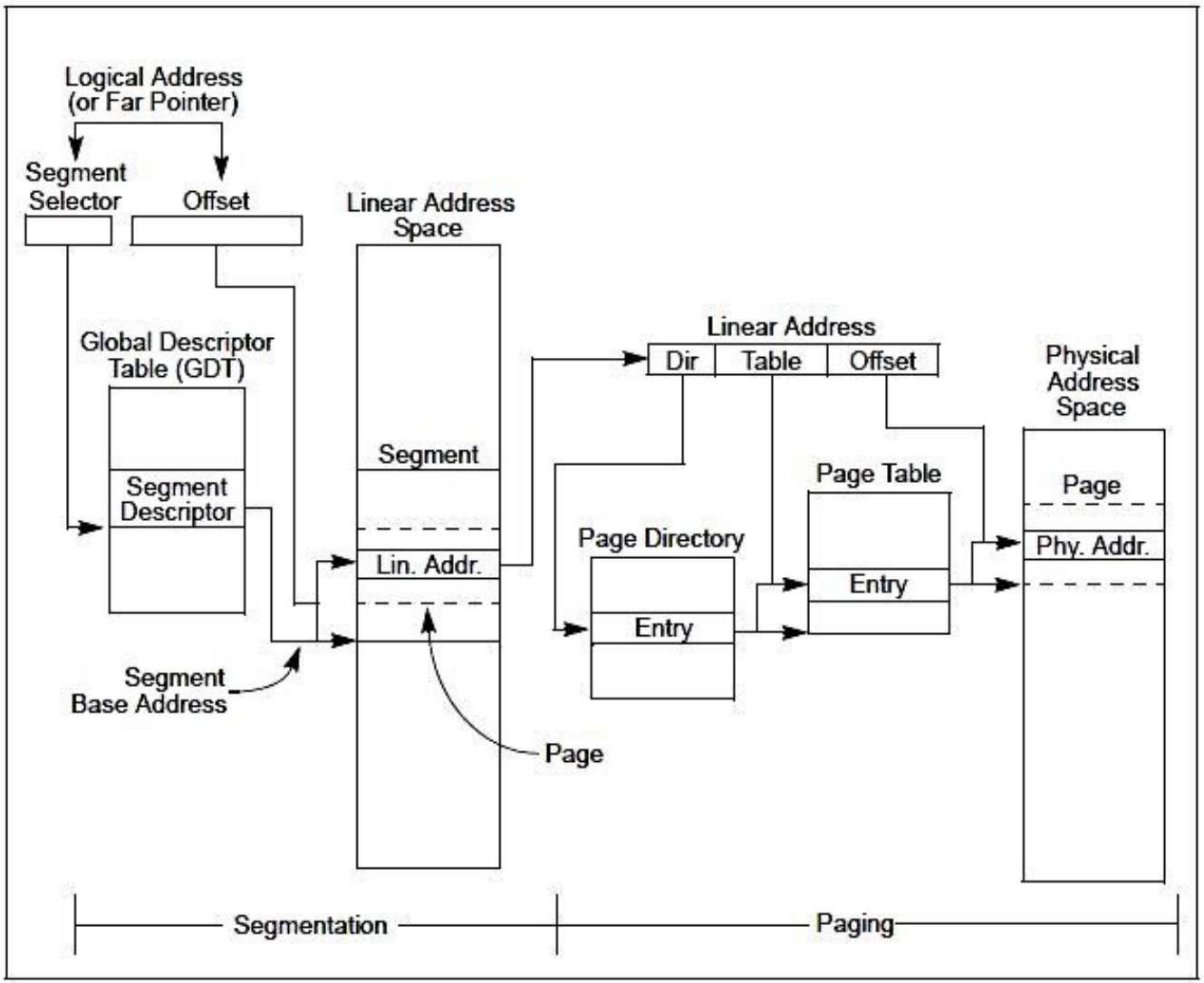
Segmentación + Paginación

Sólo para entender la interacción de las dos unidades, veamos como funciona la traducción al involucrar la unidad de segmentación.

Segmentación + Paginación

Sólo para entender la interacción de las dos unidades, veamos como funciona la traducción al involucrar la unidad de segmentación. Recuerden que al usar segmentación flat podemos olvidarnos de la unidad de segmentación al pensar la traducción de las direcciones.

Segmentación + Paginación



Casos de uso

Casos de uso

Veamos ahora algunos ejemplos de cómo el mecanismo de paginación nos permite implementar soluciones flexibles para diversos problemas.

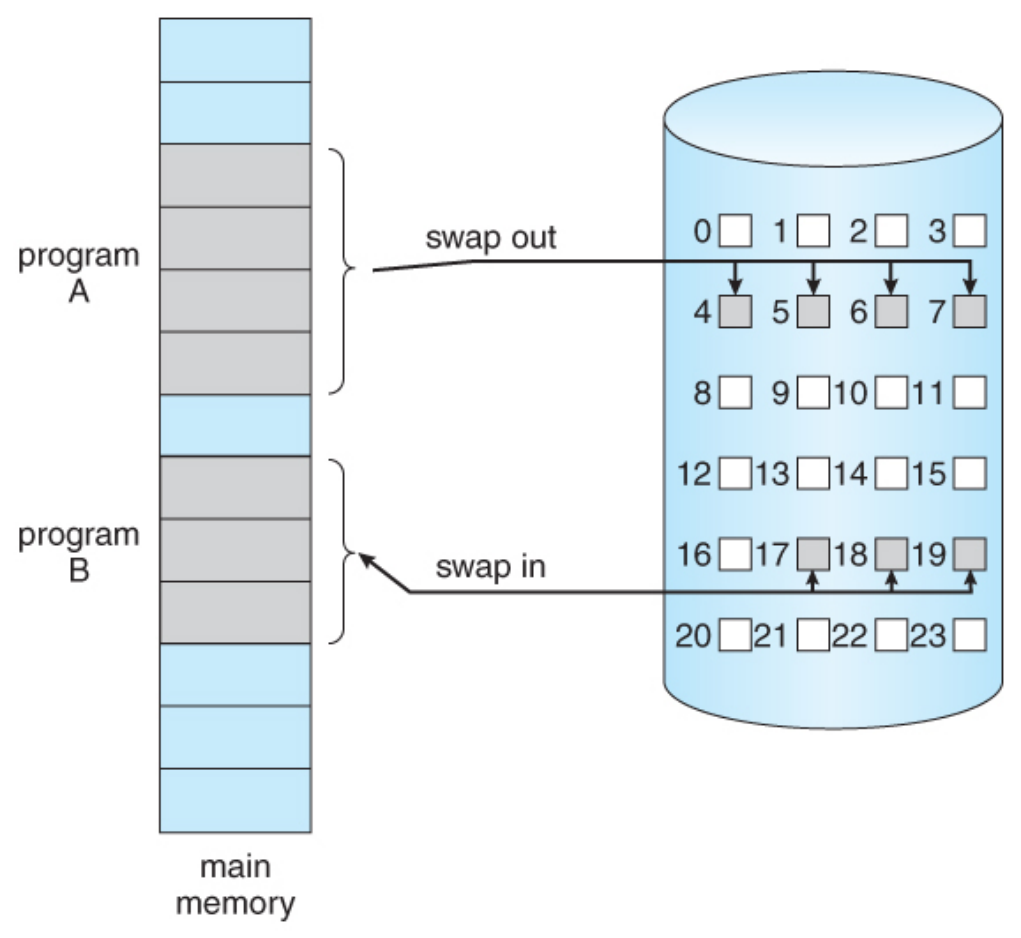
Casos de uso

Veamos ahora algunos ejemplos de cómo el mecanismo de paginación nos permite implementar soluciones flexibles para diversos problemas.
Estos son también buenos ejemplos de cómo las excepciones pueden utilizarse no sólo para detectar comportamiento patológico sino para disparar respuestas del sistema operativo frente a ciertas situaciones esperadas en el curso de su funcionamiento.

Swapping

Cuando varios procesos comparten el uso de la memoria principal y la misma se encuentra al límite de la sobre-utilización o si un proceso se encuentra suspendido mientras otro precisa ejecutarse, **el sistema operativo puede decidir mover las páginas de estos procesos desde y hacia el disco sin afectar la visión del proceso de su espacio de memoria**, incluso si al volver a volcarlo en memoria principal cambian sus direcciones físicas.

Swapping



Intercambio de procesos en memoria principal.

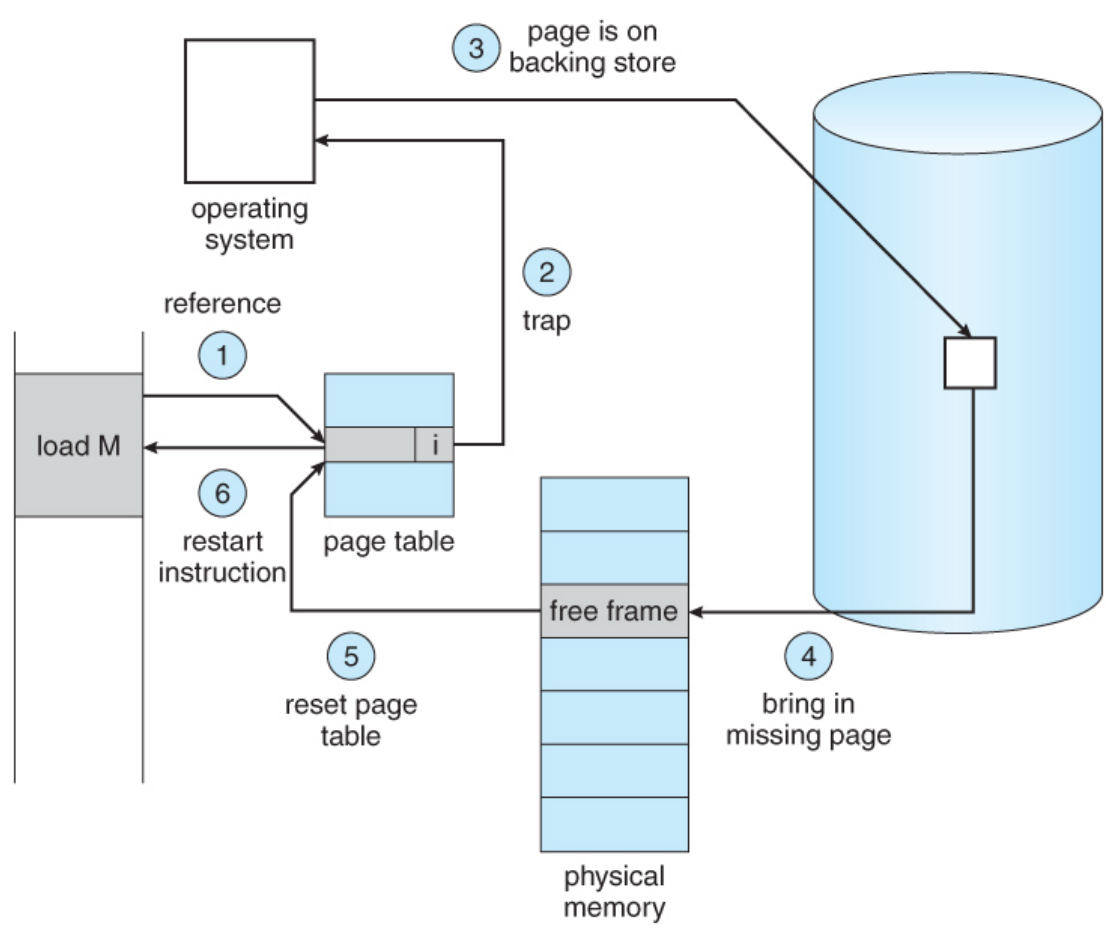
Swapping

Para implementar efectivamente este mecanismo el sistema operativo puede registrar internamente la frecuencia y prioridad de los procesos para decidir cuáles van o vuelven del disco.

Carga bajo demanda

El sistema operativo puede decidir no cargar una parte de las páginas asociadas a un proceso y marcar sus entradas como no presentes (bit **P** en 0), y luego en la rutina de atención de la excepción del **page fault** buscar la página en disco, mapearla en la tabla correspondiente, marcar el bit de presente en 1 y volver a ejecutar.

Carga bajo demanda

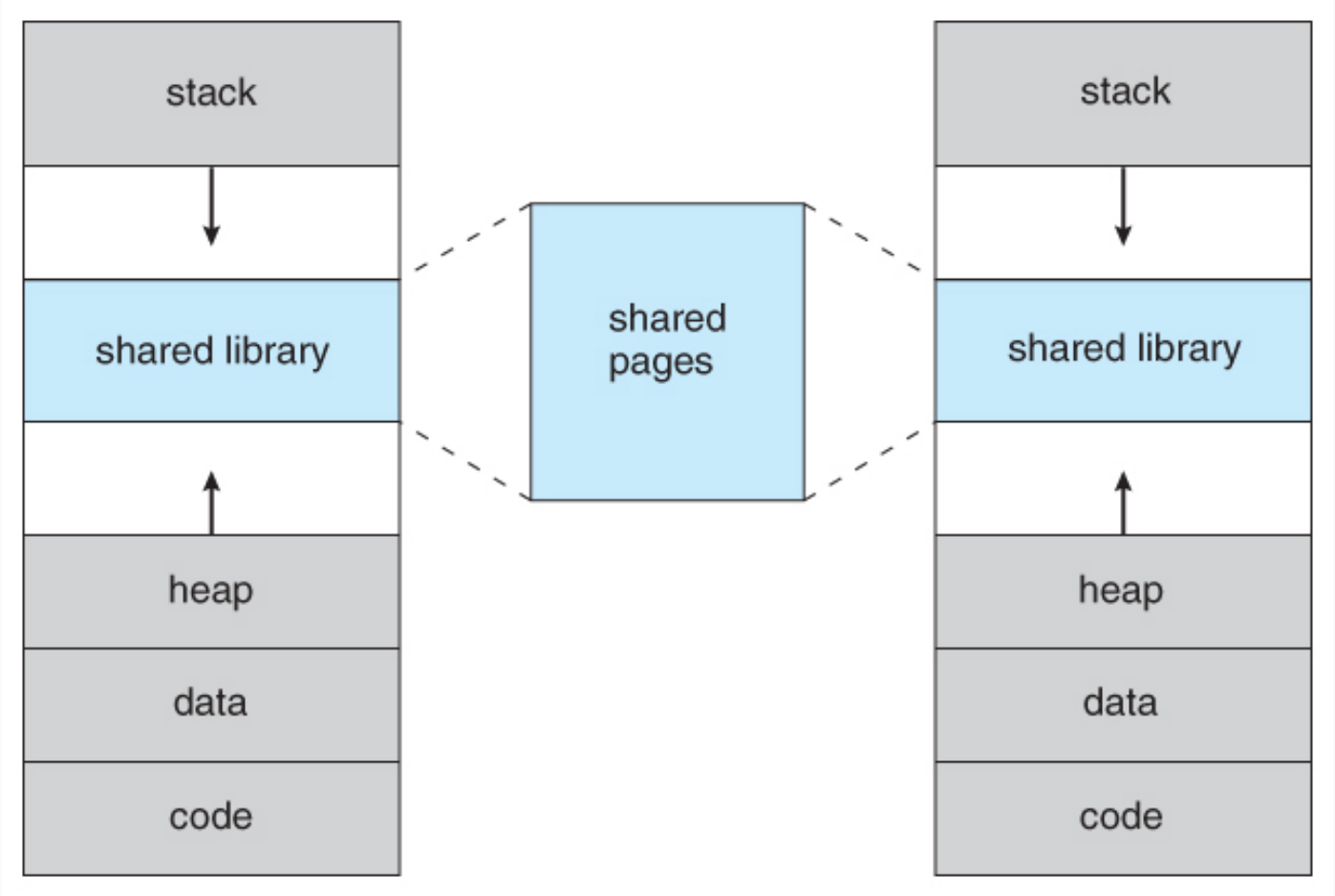


Un **page fault** no siempre es malo.

Bibliotecas compartidas

Hablamos anteriormente de las **bibliotecas compartidas**, que son utilidades que el sistema operativo expone a uno o varios procesos como forma de codigo binario que puede accederse dinámicamente. Si varios procesos acceden al mismo código no hace falta copiarlo una vez por cada instancia, podemos mapear las mismas páginas físicas (de la biblioteca compartida) en distintas estructuras por proceso.

Bibliotecas compartidas

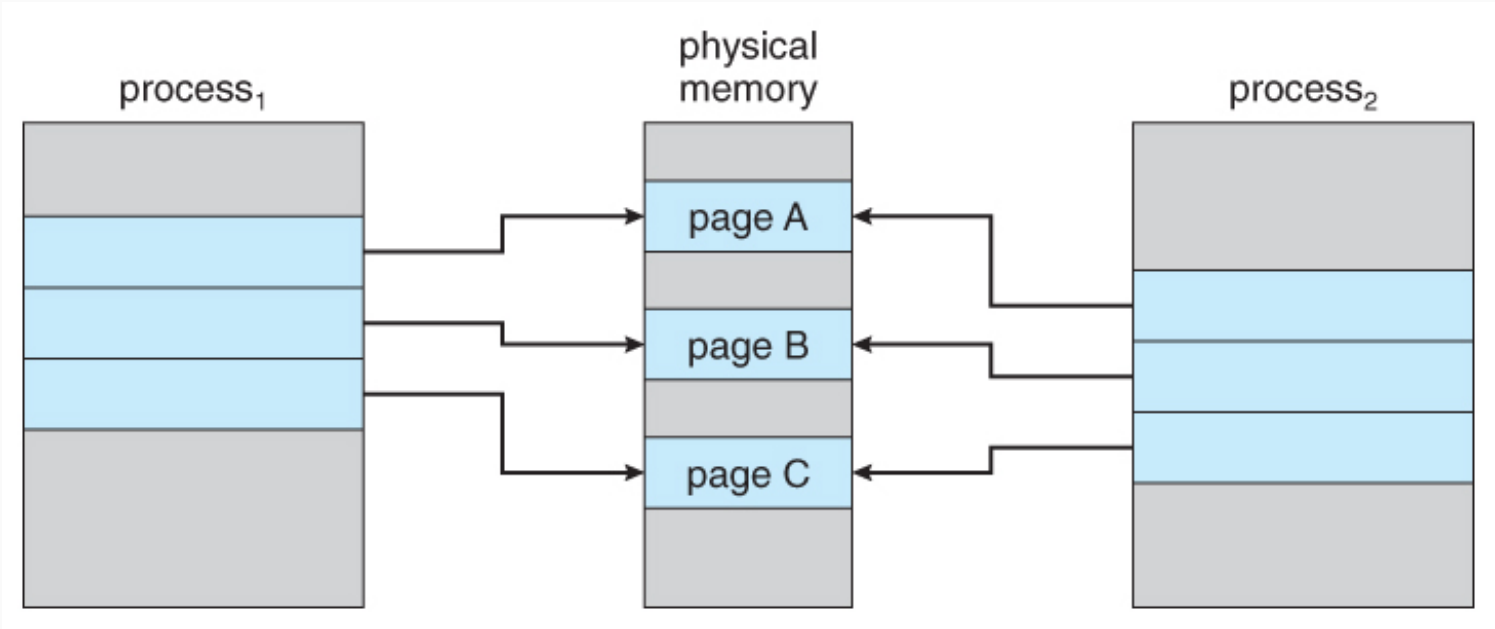


Un mismo rango físico mapeado en distintos rangos virtuales.

Copy on write

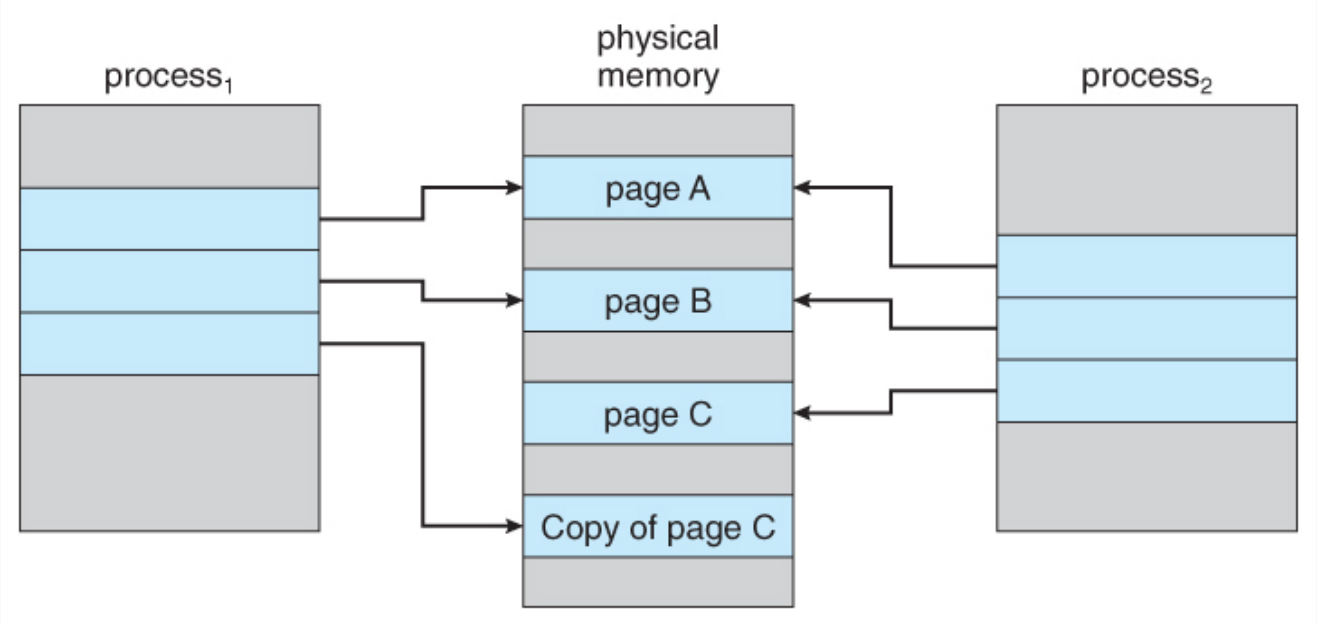
En los sistemas operativos modernos es habitual que se implemente la creación de nuevos procesos a partir de la copia del proceso en ejecución (**fork**), frente a esta situación se pueden mapear varios procesos contra el mismo conjunto de páginas físicas que contienen su código y sólo duplicarlas si se escribe en ellas y más de un proceso las está accediendo. Para esto se pueden marcar las páginas como de sólo escritura, y al detectar la excepción copia la página, la mapea como de escritura para el proceso que generó la interrupción y vuelve a ejecutar.

Copy on write



Varios procesos resultantes de un **fork** pueden compartir páginas físicas siempre y cuando no escriban en ellas.

Copy on write



Cuando un proceso escribe en una página compartida es necesario copiarla y modificar las estructuras de paginación acorde a esto.

Casos de uso

Estos son algunos casos de uso interesantes que involucran el mecanismo de **interrupciones** y el de **paginación**.

Cierre

Repaso de la introducción de hoy

En la introducción de hoy vimos:

Repaso de la introducción de hoy

En la introducción de hoy vimos:

- **Repaso de paginación como mecanismo**

Repaso de la introducción de hoy

En la introducción de hoy vimos:

- **Repaso de paginación como mecanismo**
- **Proceso de traducción de direcciones virtuales a direcciones físicas**

Repaso de la introducción de hoy

En la introducción de hoy vimos:

- **Repaso de paginación como mecanismo**
- **Proceso de traducción de direcciones virtuales a direcciones físicas**
- **Estructuras involucradas en paginación (`CR3`, `page directory`, `page table`)**

Repaso de la introducción de hoy

En la introducción de hoy vimos:

- **Repaso de paginación como mecanismo**
- **Proceso de traducción de direcciones virtuales a direcciones físicas**
- **Estructuras involucradas en paginación (`CR3`, `page directory`, `page table`)**
- **Atributos relevantes**

Repaso de la introducción de hoy

En la introducción de hoy vimos:

- **Repaso de paginación como mecanismo**
- **Proceso de traducción de direcciones virtuales a direcciones físicas**
- **Estructuras involucradas en paginación (`CR3`, `page directory`, `page table`)**
- **Atributos relevantes**
- **Casos de uso para el mecanismo de paginación**

Consultas